Cryptographic Protocols

Prof Chik How Tan
NISlab
Gjøvik University College
Chik.tan@hig.no

In this note, some of the diagrams and materials are taken from Internet

Cryptographic Protocols

- A *cryptographic protocol* is defined as a series of steps and message exchanges between multiple entities in order to achieve a specific security objective.
- Properties of protocol :
  - Everyone involved in the protocol must know the protocol and all of the steps to follow in advance
  - Everyone involved in the protocol must agree to follow it
  - The protocol must be unambiguous, that is every step is well defined and there is no chance of misunderstanding
  - The protocol must be complete, i.e. there is a specified action for every possible situation
  - It should not be possible to do or learn more than what is specified in the protocol

In this slide is taken from G. Schafer's lecture note

Applications of Cryptographic Protocols

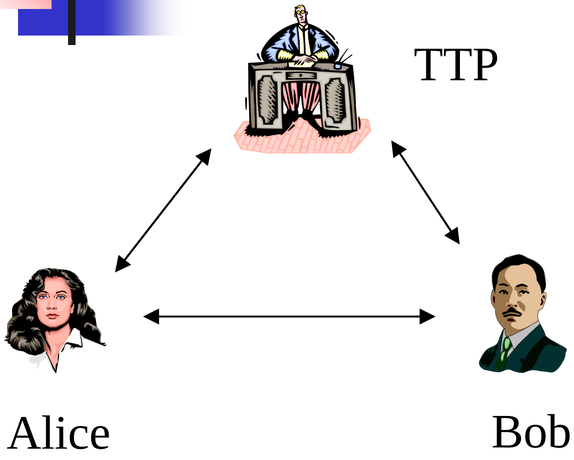
- Key exchange
- Authentication: Data origin and entity
- Authenticated key exchange
- Secret sharing
- Blind signature
- Secure elections
- Electronic money (electronic cash)
- Zero knowledge proof
- Identification

In this slide is taken from G. Schafer's lecture note

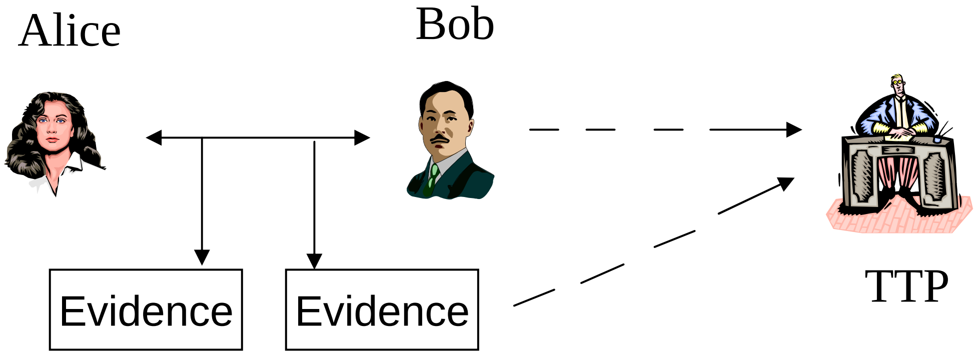
Key Exchange

- Mutual Entity Authentication: The participants believe recently replied to a specific challenge
- Mutually Authenticated Key Agreement: The protocol shall establish a fresh session key, known only to the participants in the session.
- Mutual Key Confirmation: Upon complete the protocol, both parties have seen the agreed key.
- Perfect Forward secrecy: Compromise of one key in the future will not allow to compromise any data that has been protected with other.

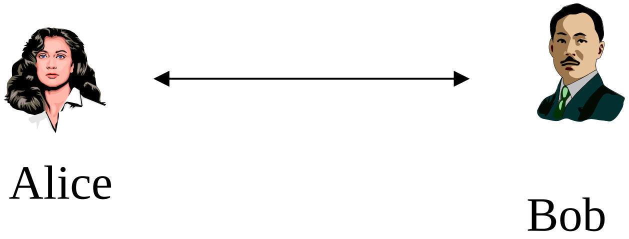
Types of Protocols



Arbitrated



Adjudicated



Self-enforcing

Types of Protocol Attacks (1)

- **Eavesdropping**: The adversary captures the information.
- **Modification:** The adversary alters the information.
- **Impersonate attack:** The adversary to act as an legitimate principal.
- **Replay:** The adversary records information and sends it in later protocol.
- **Preplay:** The adversary engage in a run of protocol prior to a run by legitimate principals.
- **Reflection:** The adversary sends protocol back to the same principal.

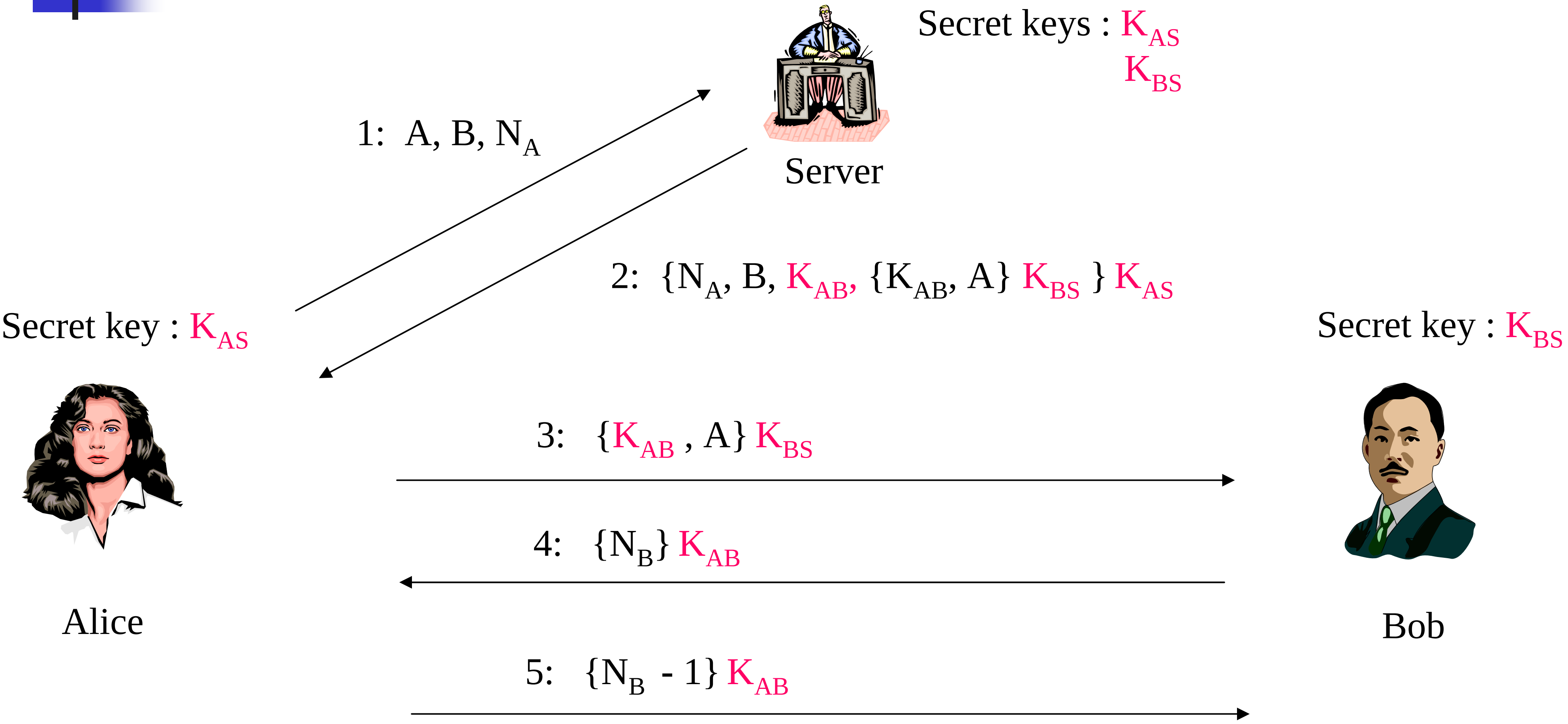
Types of Protocol Attacks (2)

- **Denial of service:** The adversary prevents legitimate principals from completing the protocol.
- **Typing Attacks**: The adversary replaces protocols message type to different types.
- **Cryptanalysis:** The adversary gain some useful leverage from protocol to help in cryptanalysis.
- **Certificate Manipulation:** The adversary chooses or modifies certificate information to attack one or more protocol run.
- **Protocol Interaction:** The adversary chooses a new protocol to interact with a known protocol.

Needham Schroeder (Share Key) Protocol

- Invented in 1978 by Roger Needham and Michael Schroeder [Nee78a]
- The protocol relies on symmetric encryption and makes use of a *trusted third party (TTP).*
- This is an important protocol. Many security analysis have been studied. Kerberos is modified from this.

Needham Schroeder (Share Key) Protocol



Server

Secret keys : K_{AS}
K_{BS}

1: A, B, N_A

2: $\{N_A, B, K_{AB}, \{K_{AB}, A\}\, K_{BS}\}\, K_{AS}$

Secret key : K_{AS}

Alice

Secret key : K_{BS}

Bob

3: $\{K_{AB}, A\}\, K_{BS}$

4: $\{N_B\}\, K_{AB}$

5: $\{N_B - 1\}\, K_{AB}$

Denning-Sacco (Replay) Attack

- If an attacker compromise an old session key, then the attacker (Ã) can impersonate A and the protocol runs as follows:

Message 1 Ã → S: A, B, N'_A

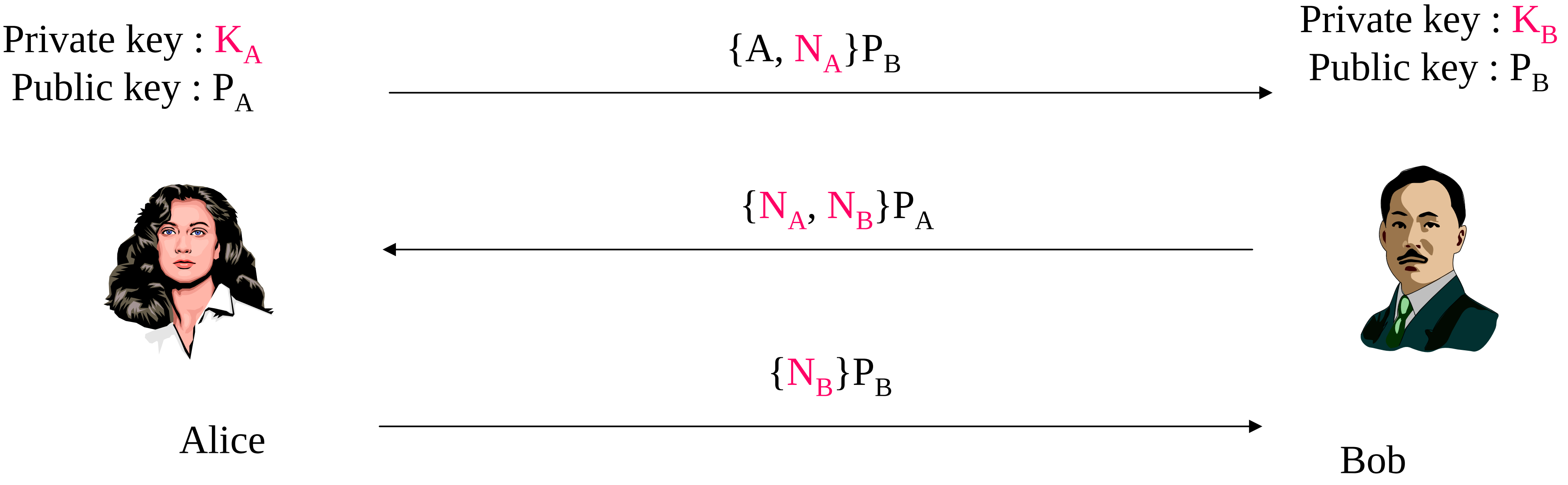
Message 2 S → Ã : $\{N'_A, B, K'_{AB}, \{K'_{AB}, A\}K_{BS}\} K_{AS}$

Message 3 Ã → B: $\{K_{AB}, A\}K_{BS}$

Message 4 B → Ã : $\{N'_B\}K_{AB}$

Message 5 Ã → B: $\{N'_B – 1\}K_{AB}$

Needham Schroeder Public Key Protocol (NSPK)



Private key : K_A
Public key : P_A

$\{A, N_A\}P_B$

$\{N_A, N_B\}P_A$

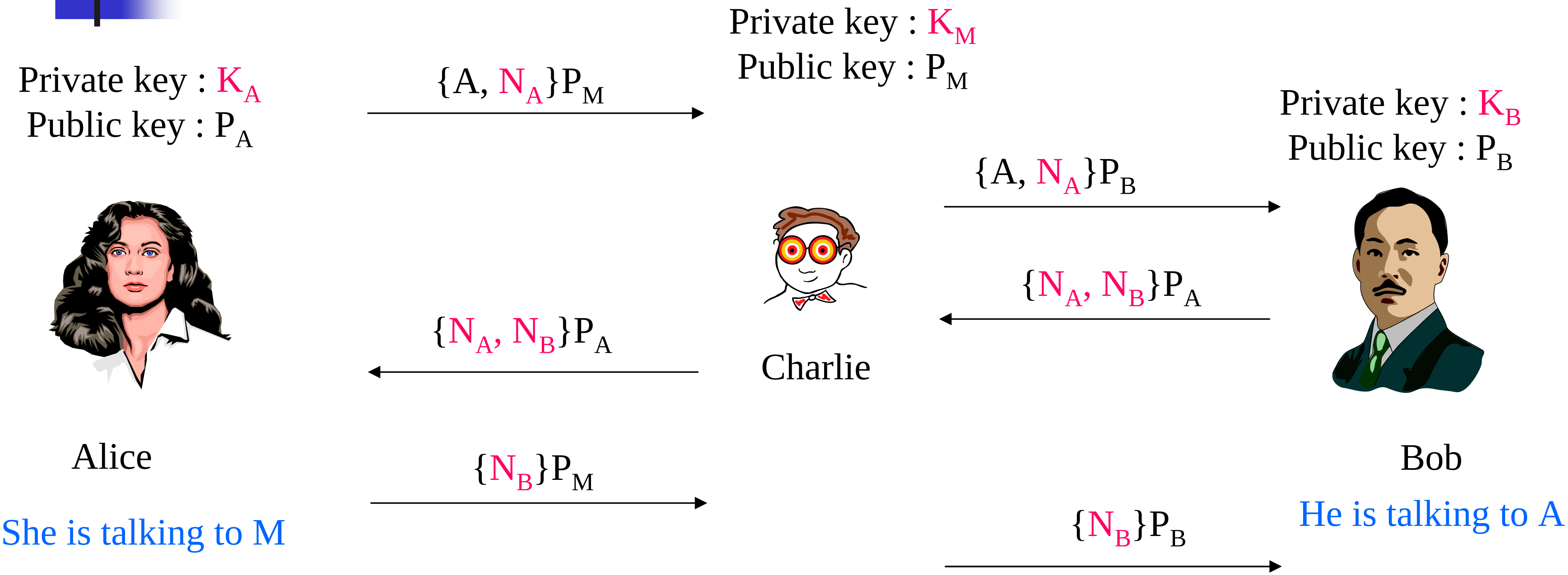
$\{N_B\}P_B$

Private key : K_B
Public key : P_B

Alice

Bob

Man-in-the-middle Attacks



Private key : K_A
Public key : P_A

Alice

She is talking to M

Private key : K_M
Public key : P_M

Charlie

Private key : K_B
Public key : P_B

Bob

He is talking to A

$\{A, N_A\}P_M$

$\{A, N_A\}P_B$

$\{N_A, N_B\}P_A$

$\{N_A, N_B\}P_A$

$\{N_B\}P_M$

$\{N_B\}P_B$

Prevent Man-in-the-middle Attacks in NSPK

Private key : K_A
Public key : P_A

Private key : K_B
Public key : P_B

$\{A, N_A\}P_B$ →

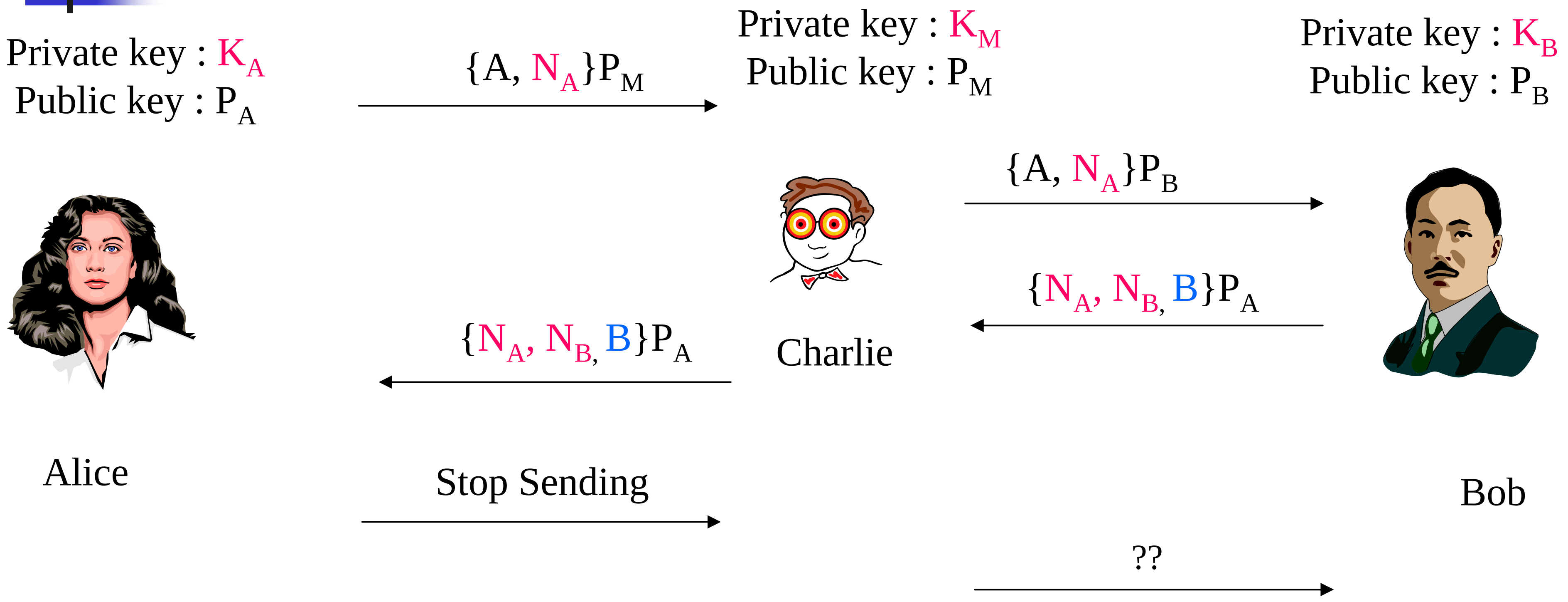
← $\{N_A, N_B, B\}P_A$

$\{N_B\}P_B$ →

Alice

Bob

Prevent Man-in-the-middle Attacks



Private key : K_A
Public key : P_A

$\{A, N_A\}P_M$ →

Private key : K_M
Public key : P_M

Private key : K_B
Public key : P_B

$\{A, N_A\}P_B$ →

$\{N_A, N_B, B\}P_A$ ←

$\{N_A, N_B, B\}P_A$ ←

Charlie

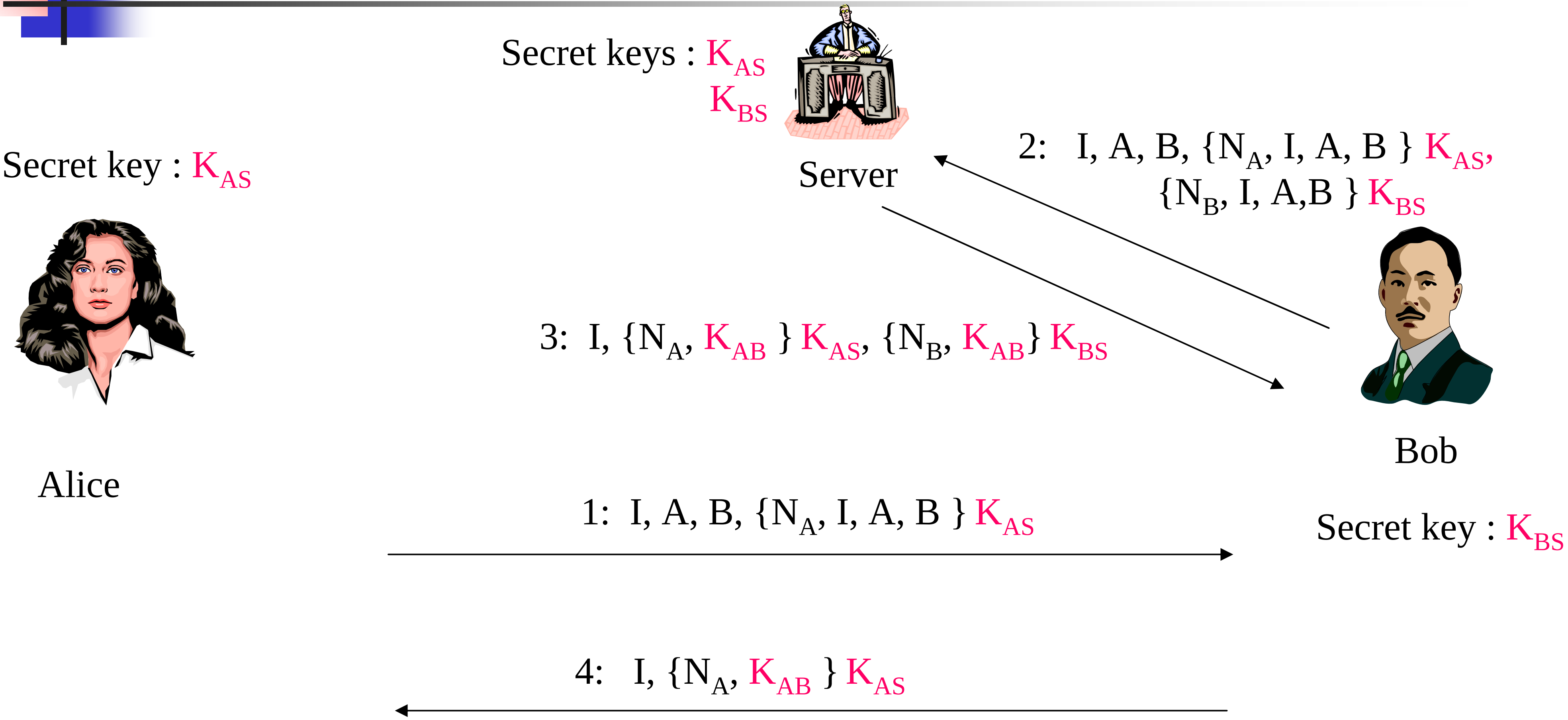
Alice

Stop Sending →

Bob

?? →

Otway-Rees Protocol



Secret keys : K_{AS} K_{BS}

Server

Secret key : K_{AS}

Alice

2: I, A, B, {N_A, I, A, B } K_{AS}, {N_B, I, A,B } K_{BS}

3: I, {N_A, K_{AB} } K_{AS}, {N_B, K_{AB}} K_{BS}

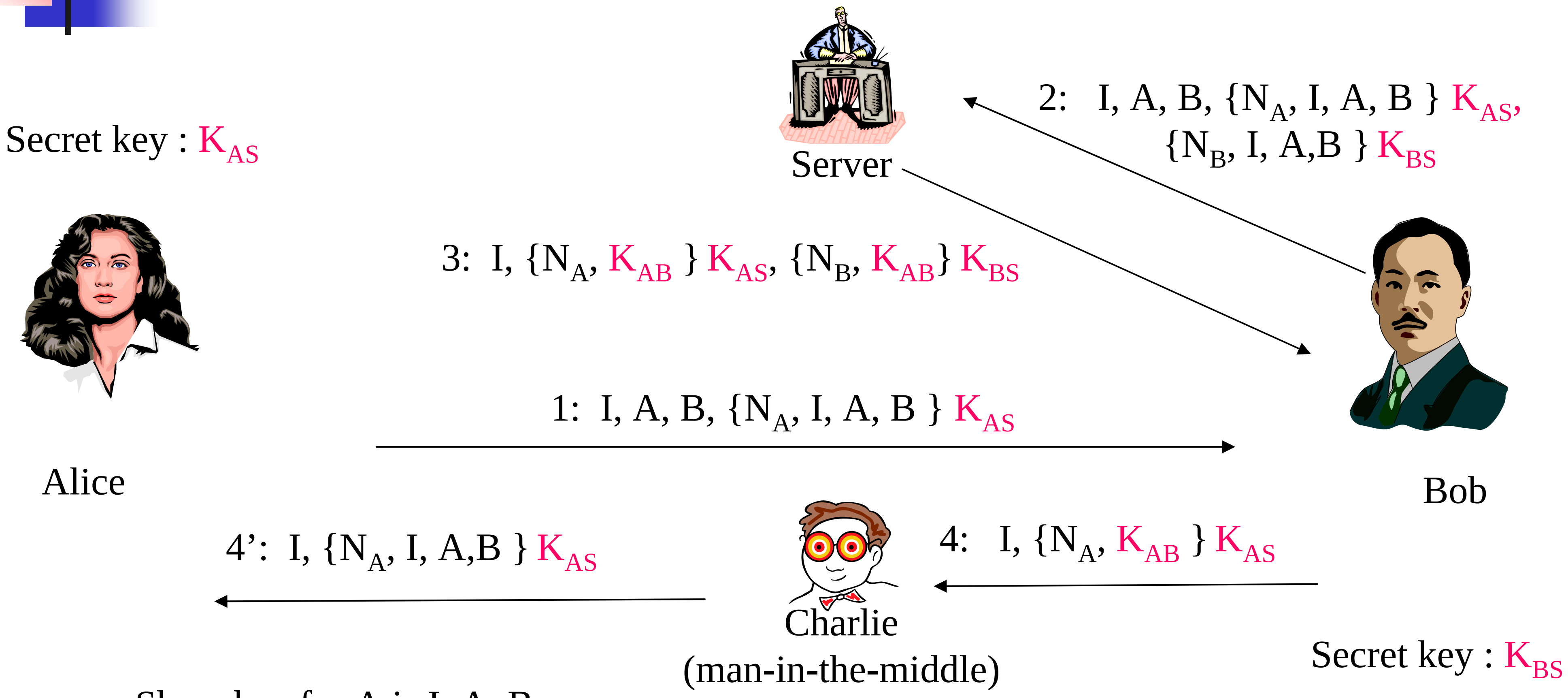
Bob

Secret key : K_{BS}

1: I, A, B, {N_A, I, A, B } K_{AS}

4: I, {N_A, K_{AB} } K_{AS}

Modification Attack on Otway-Rees Protocol



Secret key : K_{AS}

Server

2: I, A, B, $\{N_A$, I, A, B $\}$ K_{AS},
$\{N_B$, I, A,B $\}$ K_{BS}

3: I, $\{N_A$, K_{AB} $\}$ K_{AS}, $\{N_B$, $K_{AB}\}$ K_{BS}

1: I, A, B, $\{N_A$, I, A, B $\}$ K_{AS}

Alice

Bob

4’: I, $\{N_A$, I, A,B $\}$ K_{AS}

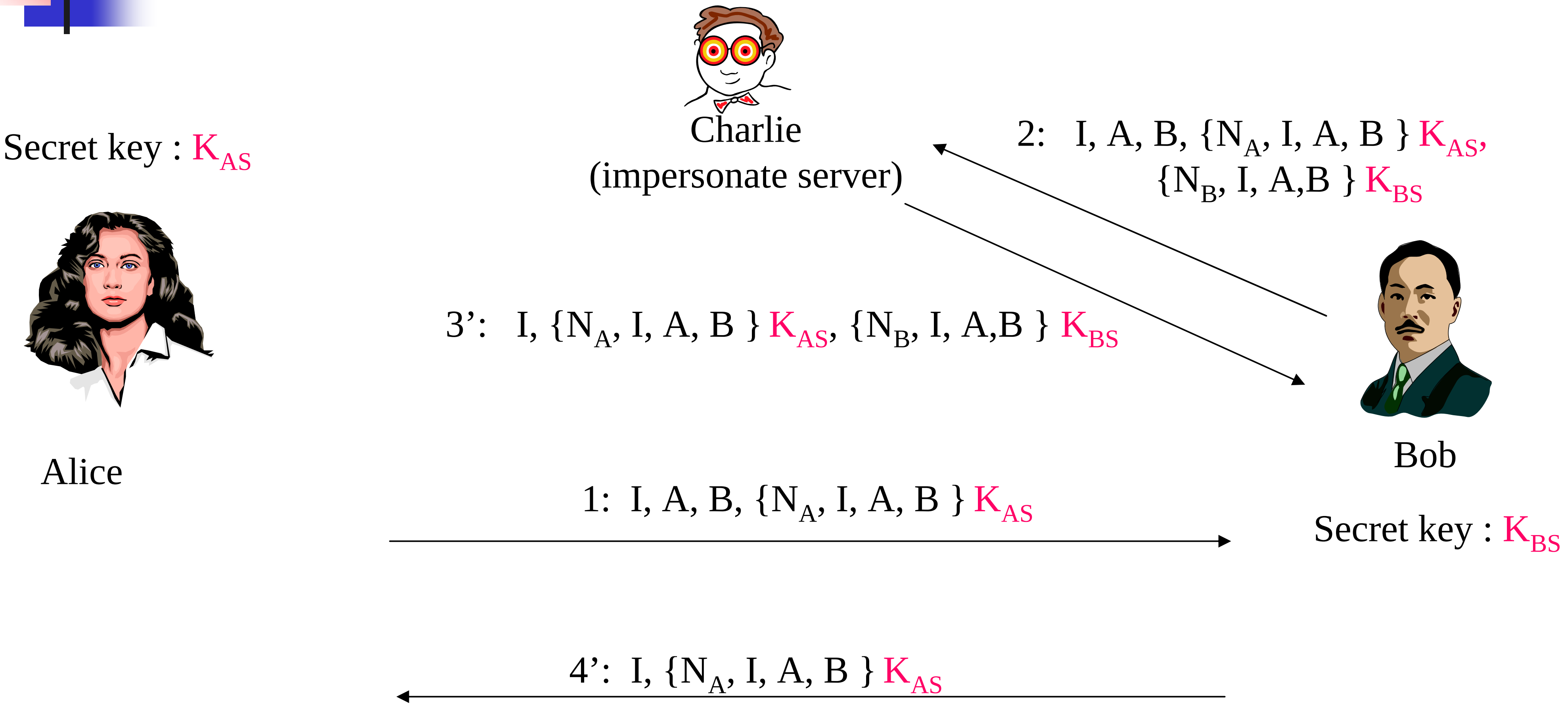
4: I, $\{N_A$, K_{AB} $\}$ K_{AS}

Charlie
(man-in-the-middle)

Secret key : K_{BS}

Share key for A is I, A, B
While B is K_{AB}

Replay or Typing Attack on Otway-Rees Protocol



Charlie
(impersonate server)

Secret key : K_{AS}

2: $I, A, B, \{N_A, I, A, B\}K_{AS}, \{N_B, I, A, B\}K_{BS}$

3': $I, \{N_A, I, A, B\}K_{AS}, \{N_B, I, A, B\}K_{BS}$

Alice

Bob

1: $I, A, B, \{N_A, I, A, B\}K_{AS}$

Secret key : K_{BS}

4': $I, \{N_A, I, A, B\}K_{AS}$

Kerberos (Version 4)

- Kerberos is an authentication and access control service for workstation clusters that was designed at the MIT during the late 1980s
- Design goals:
  - *Security:* eavesdroppers or active attackers should not be able to obtain the necessary information to impersonate a user when accessing a service
  - *Reliability:* as every use of a service requires prior authentication, Kerberos should be highly reliable and available
  - *Transparency:* the authentication process should be transparent to the user beyond the requirement to enter a password
  - *Scalability:* the system should be able to support a large number of clients and servers

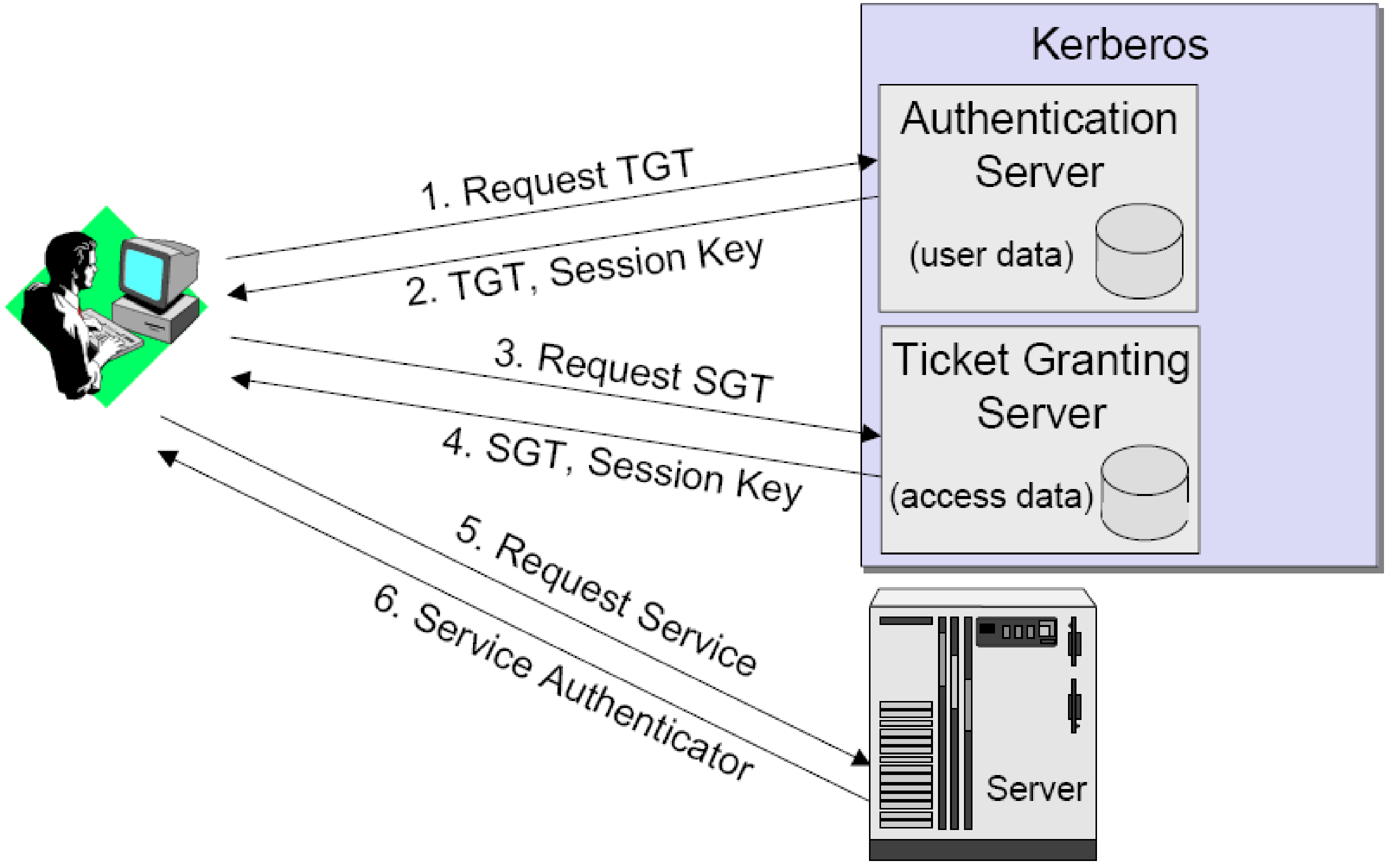
In this slideis taken from G. Schafer's lecture note

Kerberos (Version 4) (Cont'd)

- *Authentication:* Alice will authenticate to an *authentication server (AS)* who will provide a temporal permit to demand access for services. This permit is called *ticket-granting ticket (*$Ticket_{TGS}$*)* and is comparable to a temporal passport.
- *Access control:* by presenting her $Ticket_{TGS}$ Alice can demand a *ticket granting server (TGS)* to obtain access for a service provided by a specific server *S1*. The TGS decides if the access will be permitted and answers with a service granting ticket $Ticket_{S1}$ for server *S1*.
- *Key exchange:* the authentication server provides a session key for communication between Alice and TGS and the TGS provides a session key for communication between Alice and *S1*. The use of these session keys also serves for authentication purposes.

In this slide is taken from G. Schafer's lecture note

Kerberos (Version 4)



In this slide is taken from G. Schafer's lecture note

Kerberos (Version 4)

- The user logs on his workstation and requests to access a service:
- The user first sends his name, the name of an appropriate ticket granting server *TGS* and a timestamp t_A to the authentication server *AS* through a workstation

  1.) $A \rightarrow AS: (A, TGS, t_A)$

- The AS verifies, that A may authenticate itself to access services, generates the key K_A out of A's password (which is known to him), extracts the workstation address *AddrA* of the request, creates a ticket granting ticket *TicketTGS* and a session key $K_{A,TGS}$, and sends the following message to A:

  2.) $AS \rightarrow A: \{K_{A,TGS}, TGS, t_{AS}, LifetimeTicket_{TGS}, Ticket_{TGS}\}K_A$

  with $Ticket_{TGS} = \{K_{A,TGS}, A, AddrA, TGS, t_{AS}, LifetimeTicket_{TGS}\}K_{AS,TGS}$

- Upon receipt of this message, the workstation asks Alice to type in her password, computes the key K_A from it, and uses this key to decrypt the message. If Alice does not provide her "authentic" password, the extracted values will be "garbage" and the rest of the protocol will fail

In this slide is taken from G. Schafer's lecture note

Kerberos (Version 4)

- Alice creates a so-called *authenticator* and sends it together with the ticket-granting ticket and the name of server S1 to TGS:

3.) $A \rightarrow TGS: (S1, Ticket_{TGS}, Authenticator_{A,TGS})$

with $Authenticator_{A,TGS} = \{A, AddrA, t'_A\}K_{A,TGS}$

- Upon receipt, *TGS* decrypts $Ticket_{TGS}$, extracts the key $K_{A,TGS}$ from it and uses this key to decrypt $Authenticator_{A,TGS}$. If the name and address of the authenticator and the ticket are matching and the timestamp t'_A is still fresh, it checks if A may access the service S1 and creates the following message:

4.) $TGS \rightarrow A: \{K_{A,S1}, S1, t_{TGS}, Ticket_{S1}\}K_{A,TGS}$

with $Ticket_{S1} = \{K_{A,S1}, A, AddrA, S1, t_{TGS}, LifetimeTicket_{S1}\}K_{TGS,S1}$

- Alice decrypts the message and does now hold a session key for secure communication with *S1*. She now sends a message to *S1* to show him her ticket and a new authenticator:

5.) $A \rightarrow S1: (Ticket_{S1}, Authenticator_{A,S1})$ with $Authenticator_{A,S1} = \{A, AddrA, t''_A\}K_{A,S1}$

In this slide is taken from G. Schafer's lecture note

Kerberos (Version 4)

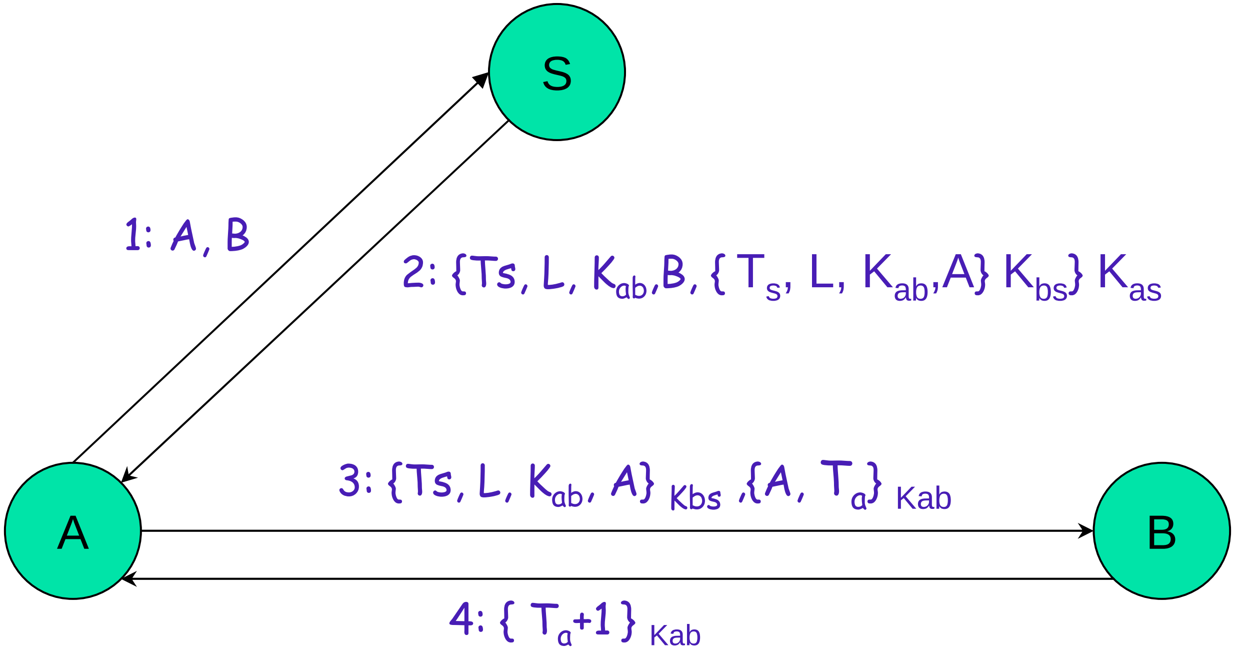
- Upon receipt, *S1* decrypts the ticket with the key $K_{TGS,S1}$ he shares with *TGS* and obtains the session key $K_{A,S1}$ for secure communication with A. Using this key he checks the authenticator and responds to A:

6.) $S1 \rightarrow A: \{t''_A + 1\}K_{A,S1}$

- By decrypting this message and checking the contained value, Alice can verify, that she is really communicating with *S1*, as only he (besides *TGS*) knows the key $K_{TGS,S1}$ to decrypt $Ticket_{S1}$ which contains the session key $K_{A,S1}$, and so only he is able to decrypt $Authenticator_{A,S1}$ and to answer with $t''_A + 1$ encrypted with $K_{A,S1}$
- The protocol described above is the Kerberos Version 4 dialogue. A number of deficiencies have been found in this protocol, so a new Version 5 of the protocol has been defined

In this slide is taken from G. Schafer's lecture note

Kerberos (Version 4)



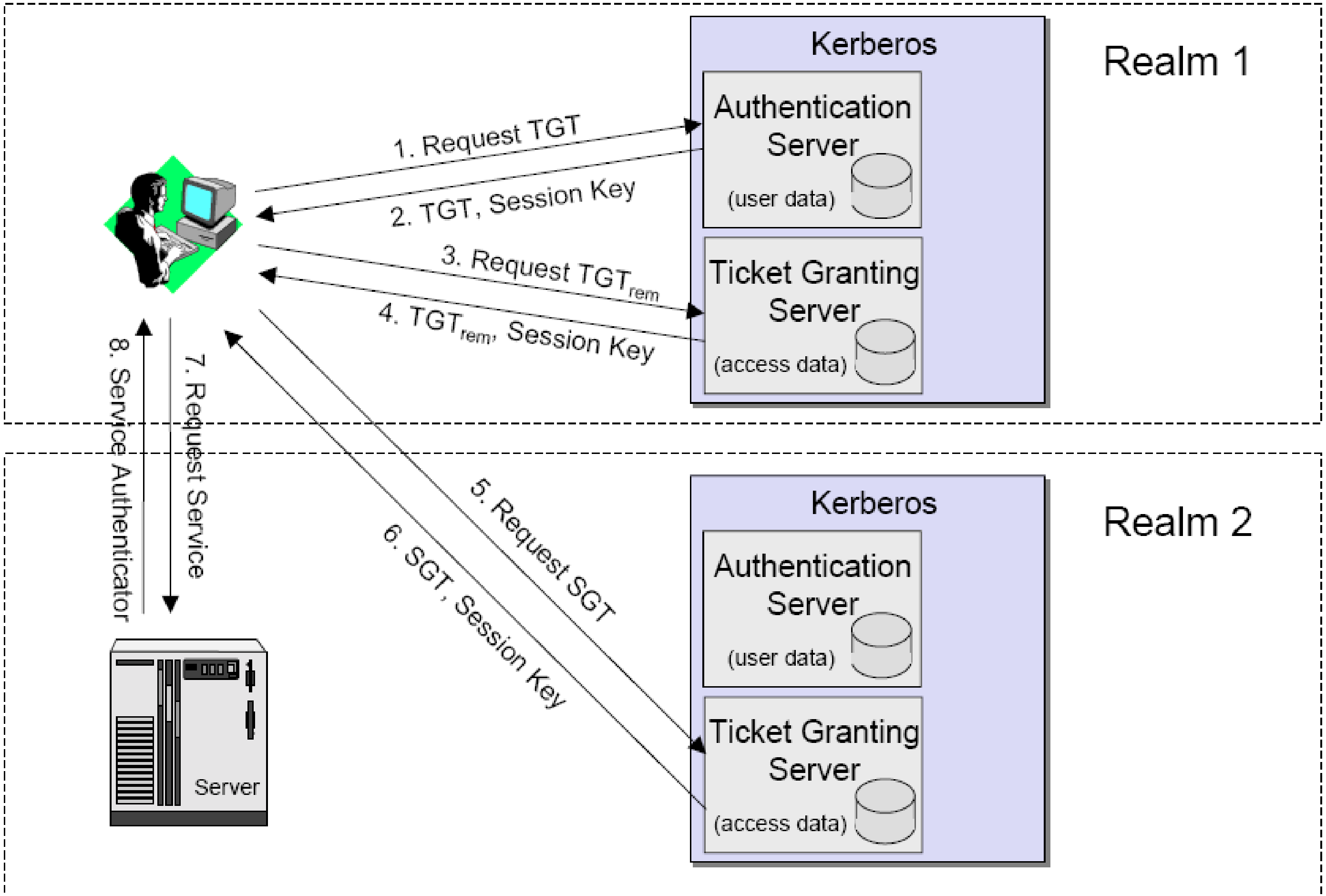
1: A, B

2: $\{T_s, L, K_{ab}, B, \{T_s, L, K_{ab}, A\}\, K_{bs}\}\, K_{as}$

3: $\{T_s, L, K_{ab}, A\}_{Kbs}, \{A, T_a\}_{Kab}$

4: $\{T_a+1\}_{Kab}$

Multiple Domain Kerberos



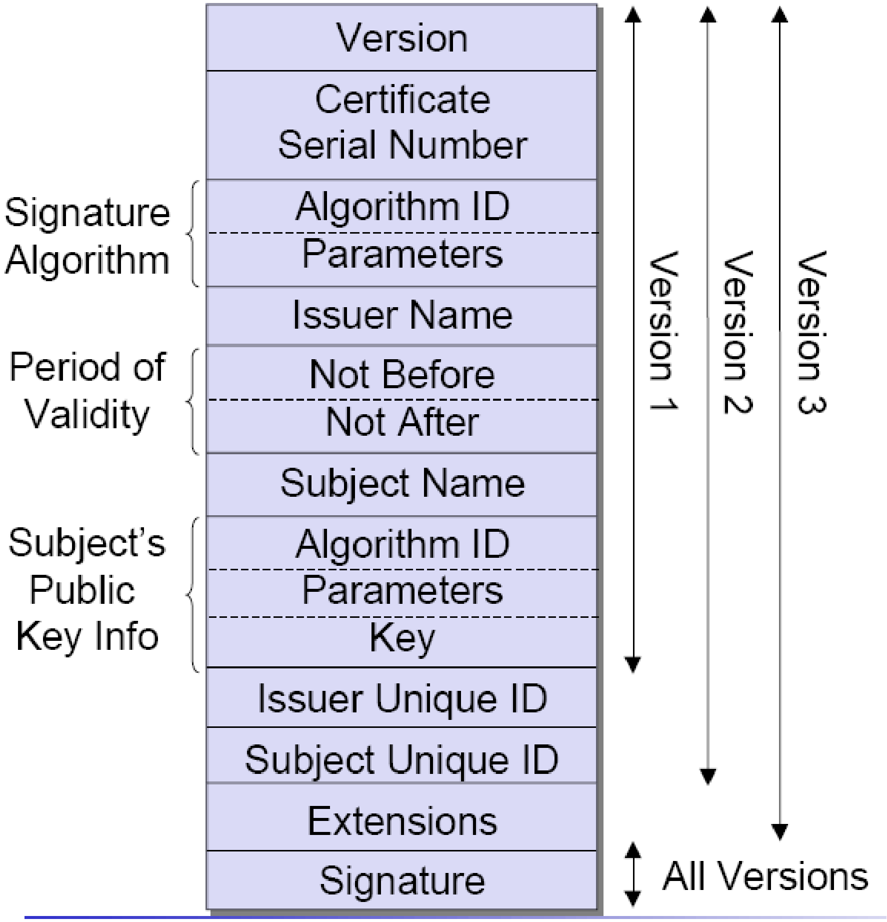
In this slide is taken from G. Schafer's lecture note

X.509 – Public Key Certificate

- X.509 is an international recommendation of ITU-T and is part of the X.500-series defining directory services:
  - The first version of X.509 was standardized in 1988
  - A second version standardized 1993 resolved some security concerns
  - A third version was drafted in 1995
- X.509 defines a framework for provision of authentication services, comprising:
  - Certification of public keys and certificate handling:
  - Certificate format
  - Certificate hierarchy
  - Certificate revocation lists
- Three different dialogues for direct authentication:
  - One-way authentication, requires synchronized clocks
  - Two-way mutual authentication, still requires synchronized clocks
  - Three-way mutual authentication entirely based on random numbers

In this slide is taken from G. Schafer's lecture note

X.509 – Public Key Certificate


Version

Certificate Serial Number

Signature Algorithm
- Algorithm ID
- Parameters

Issuer Name

Period of Validity
- Not Before
- Not After

Subject Name

Subject's Public Key Info
- Algorithm ID
- Parameters
- Key

Issuer Unique ID

Subject Unique ID

Extensions

Signature

- Certificate certifies a public key belong to specific name
- A Certificate is issued by a certificate authority
- Knowing the public key of CA, anyone can verify the certificate
- Certificate can avoid on-line participation of TTP
- The security of private key of CA is crucial to the security of all users

In this slide is taken from G. Schafer's lecture note

Reflection Attacks

Protocol A: A and B has a share key k

1. **A** → **B** : $\{N_A\}k$
2. **B** → **A** : $\{N_B\}k; N_A$
3. **A** → **B** : N_B

1. **A** → **C** : $\{N_A\}k$

1’. **C** → **A** : $\{N_A\}k$

2’. **A** → **C** : $\{N'_A\}k; N_A$

2. **C** → **A** : $\{N'_A\}k; N_A$

3. **A** → **C** : N'_A

3’. **C** → **A** : N'_A

Denial of Service Attacks (1)

Protocol B:

1. **A** → **B** : g^x;

2. **B** → **A** : Cert (B); g^y; $E_K(sig_B(g^x, g^y))$

3. **A** → **B** : Cert (A); $E_K(sig_A(g^x, g^y))$

$K = g^{xy}$

Denial of Service Attacks (2)

Protocol C:

1. **A** → $\mathbf{M_B}$: g^x;
2. $\mathbf{M_B}$ → **B** : g^x;
3. **B** → $\mathbf{M_B}$: Cert (B); g^y; $E_K(sig_B(g^x, g^y))$
4. $\mathbf{M_B}$ → **A** : Cert (B); g^y; $E_K(sig_B(g^x, g^y))$
5. **A** → $\mathbf{M_B}$: Cert (A); $E_K(sig_A(g^x, g^y))$

A: A ⇔ B B: M ⇔ B is not a complete protocol

Certificate Manipulation

Protocol D:

1. **A** → **B** : g^x; Cert(A)
2. **B** → **A** : g^y; Cert(B)

Public key of A, B, C are
g^a , g^b and g^{ac}
Share key $K_{AB}=g^{ay+bx}$

1. **A** → $\mathbf{C_B}$: g^x; Cert(A)

1’. **C** → **B** : g^x; Cert(C)
2’. **B** → **C** : g^y; Cert(B)
2. $\mathbf{C_B}$ → **A** : g^{yc}; Cert(B)

Share key $K_{AB}=g^{ayc+bx}$
$K_{CB}=g^{ayc+bx}$

Interleaving Attack on NSPK

Protocol E:

Message 1 $A \rightarrow E$: $\{A, N_a\}K_e$

i $E_A \rightarrow B$: $\{A, N_a\}K_b$

ii $B \rightarrow E_A$: $\{N_a, N_b\}K_a$

Message 2 $E \rightarrow A$: $\{N_a, N_b\}K_a$

Message 3 $A \rightarrow E$: $\{N_b\}K_e$

iii $E_A \rightarrow B$: $\{N_b\}K_b$